

(5) Multivariate Deskriptivstatistik

Ziel dieser Sitzung ist es, zu verdeutlichen, wie die Berechnung multivariater Deskriptivstatistiken implementiert wird, und sich der Grundlagen der Frequentistischen Inferenz zu besinnen.

Multivariate Deskriptivstatistiken

Wir simulieren zunächst einen Datensatz von $n = 30$ vierdimensionalen Messwerten ($m = 4$).

```
# R-Pakete
library(MASS) # multivariate Normalverteilungen
library(matrixcalc) # Matrix-Paket (is.positive.definite())

# Simulationsparameter
set.seed(1) # reproduzierbare Randomisierung
m = 4 # Datenpunktdimension
n = 30 # Anzahl Realisierungen
mu = rep(0,m) # Erwartungswertparameter
Sigma = matrix(runif(m^2), nrow = m) # zufällige Matrix
Sigma = 0.5*(Sigma+t(Sigma)) # symmetrische Matrix
Sigma = Sigma + m*diag(m) # positiv definite Matrix

# Datensatzgeneration
Y = t(mvrnorm(n,mu,Sigma)) # Y = (y_1, ..., y_n) \sim N(\mu, \Sigma)
fname = "Multivariate_Deskriptivstatistik.csv" # Dateiname
write.csv(Y, # Datenspeichern
  file = fname,
  row.names = FALSE)
```

Wir laden nun die Datenmatrix und berechnen auf Grundlage des in der Vorlesung diskutierten Theorems zu Datenmatrix und multivariate Deskriptivstatistiken das Stichprobenmittel, die Stichprobenkovarianzmatrix und die Stichprobenkorrelationsmatrix.

```
# Laden einer m x n Datenmatrix
fname = "Multivariate_Deskriptivstatistik.csv"
Y = as.matrix(read.table(fname, sep = ",", header = TRUE))

# Deskriptivstatistiken
n = ncol(Y) # Anzahl Datenvektorealisationen
I_n = diag(n) # Einheitsmatrix I_n
J_n = matrix(rep(1,n^2), nrow = n) # 1_{nn}
y_bar = (1/n)*Y %>% J_n[,1] # Stichprobenmittel
C = (1/(n-1))*(Y %>% (I_n-(1/n)*J_n) %>% t(Y)) # Stichprobenkovarianzmatrix
D = diag(1/sqrt(diag(C))) # Kov-Korr-Transformationsmatrix
R = D %>% C %>% D # Stichprobenkorrelationsmatrix

# Ausgabe
print(y_bar)
```

```
      [,1]
[1,] -0.20454286
[2,]  0.40088096
[3,] -0.31740484
[4,] -0.08424694
```

```
print(C)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,]  3.9134127 -0.2422984  1.2068574  1.5554252
[2,] -0.2422984  2.9762937 -0.3353690  0.3849835
[3,]  1.2068574 -0.3353690  3.6424652  0.6982073
[4,]  1.5554252  0.3849835  0.6982073  3.4603723
```

```
print(R)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] 1.00000000 -0.07099616  0.3196542  0.4226782
[2,] -0.07099616  1.00000000 -0.1018562  0.1199617
[3,]  0.31965422 -0.10185616  1.0000000  0.1966642
[4,]  0.42267824  0.11996174  0.1966642  1.0000000
```

Wir überprüfen die Richtigkeit dieser Ergebnisse mithilfe der R-Funktionen `rowMeans` (bzw. `colMeans`), `cov` und `cor`.

```
# Deskriptivstatistiken
y_bar = as.matrix(rowMeans(Y))           # Strichprobenmittel als Zeilenmittel von Y
y_bar = as.matrix(colMeans(t(Y)))       # Strichprobenmittel als Spaltenmittel von Y^T
C      = cov(t(Y))                       # Stichprobenkovarianzmatrix
R      = cor(t(Y))                       # Stichprobenkorrelationsmatrix

# Ausgabe
print(y_bar)
```

```
      [,1]
[1,] -0.20454286
[2,]  0.40088096
[3,] -0.31740484
[4,] -0.08424694
```

```
print(C)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,]  3.9134127 -0.2422984  1.2068574  1.5554252
[2,] -0.2422984  2.9762937 -0.3353690  0.3849835
[3,]  1.2068574 -0.3353690  3.6424652  0.6982073
[4,]  1.5554252  0.3849835  0.6982073  3.4603723
```

```
print(R)
```

```
      [,1]      [,2]      [,3]      [,4]
[1,] 1.00000000 -0.07099616  0.3196542  0.4226782
[2,] -0.07099616  1.00000000 -0.1018562  0.1199617
[3,]  0.31965422 -0.10185616  1.0000000  0.1966642
[4,]  0.42267824  0.11996174  0.1966642  1.0000000
```

Schließlich berechnen wir die quadrierte Euklidische Distanz und die Mahalanobis-Distanz des berechneten Stichprobenmittels vom Nullpunkt basierend auf der Stichprobenkovarianzmatrix.

```
# Laden einer m x n Datenmatrix
fname = "Multivariate_Deskriptivstatistik.csv"
Y      = as.matrix(read.table(fname, sep = ",", header = TRUE))

# Deskriptivstatistiken
n      = ncol(Y)                       # Anzahl Datenvektorealisierungen
I_n    = diag(n)                       # Einheitsmatrix I_n
J_n    = matrix(rep(1,n^2), nrow = n)  # 1_{nn}
y_bar  = (1/n)*Y %>% J_n[,1]           # Stichprobenmittel
C      = (1/(n-1))*(Y %>% (I_n-(1/n)*J_n) %>% t(Y)) # Stichprobenkovarianzmatrix

# Ausgabe
cat("quadrierte Euklidische Distanz von 0_4 : ", t(y_bar) %>% y_bar,
    "\nMahalanobis-Distanz von 0_4          : ", t(y_bar) %>% solve(C) %>% y_bar, "\n")
```

```
quadrierte Euklidische Distanz von 0_4 : 0.3103867
Mahalanobis-Distanz von 0_4          : 0.07736309
```

Grundlagen Frequentistischer Inferenz

Anhand untenstehender Simulation der Dualität von Konfidenzintervallen und Hypothesentests im Szenario des (univariaten) Einstichproben-T-Tests wiederholen wir noch einmal die intuitiven Grundlagen der Frequentistischen Inferenz.

```
# Modellformulierung
n       = 12                               # Stichprobengröße
mu      = 2                               # wahrer, aber unbekannter Erwartungswertparameter
sigsqr  = 1                               # wahrer, aber unbekannter Varianzparameter

# Konfidenzintervallparameter und Testparameter
delta   = 0.95                             # Konfidenzbedingung
t_delta = qt((1+delta)/2, n-1)             # \Psi^{-1}((\delta + 1)/2, n-1)
mu_0    = mu                               # Nullhypothesenparameter

# Simulationen
set.seed(1)                                # reproduzierbare Randomisierung
ns      = 1e5                              # Anzahl Simulationen
y_bar   = rep(NaN,ns)                      # Stichprobenmittelarray
s       = rep(NaN,ns)                      # Stichprobenstandardabweichungarray
kappa   = matrix(rep(NaN,2*ns), ncol = 2)  # Konfidenzintervallarray
kfn     = rep(NaN,ns)                      # Überdeckungsindikatorarray
phi     = rep(NaN,ns)                      # Testarray
for(i in 1:ns){                            # Simulationsiterationen

  # Stichprobenrealisation und Konfidezintervallevaluation
  y       = rnorm(n,mu_0,sqrt(sigsqr))      # Stichprobenrealisierung
  y_bar[i] = mean(y)                       # Stichprobenmittel
  s[i]    = sd(y)                          # Stichprobenstandardabweichung
  kappa[i,1] = y_bar[i] - (s[i]/sqrt(n))*t_delta # untere KI Grenze
  kappa[i,2] = y_bar[i] + (s[i]/sqrt(n))*t_delta # obere KI Grenze

  # Überdeckungs- und Testevaluation
  if(kappa[i,1] <= mu_0 & mu_0 <= kappa[i,2]){
    kfn[i] = 1} else{kfn[i] = 0}            # Überdeckungsindikatorevaluation
  if(kappa[i,1] <= mu_0 & mu_0 <= kappa[i,2]){
    phi[i] = 0} else{phi[i] = 1}           # Testevaluation

# Ausgabe
cat(" geschätztes Konfidenzniveau : ", mean(kfn),
    "\ngeschätzter Testumfang      : ", mean(phi))
```

```
geschätztes Konfidenzniveau : 0.9507
geschätzter Testumfang      : 0.0493
```

Wir visualisieren die ersten 100 Simulationen als Abbildung 1.

```
# Visualisierungsparameter
library(latex2exp)
par(
  family      = "sans",
  bty         = "l",
  mfcol       = c(2,1),
  lwd         = 1,
  las         = 1,
  mgp         = c(2,1,0),
  xaxs        = "i",
  yaxs        = "i",
  font.main   = 1,
  cex         = 1,
  cex.main    = 1)

# Konfidenzintervallberechnung
nv           = 100                                # Simulationsanzahl für Visualisierung
y_bar       = y_bar[1:nv]                        # Selektion der ersten nv Stichprobenmittel
phi         = phi[1:nv]                          # Selektion der ersten nv Testwertsimulationen
kappa       = kappa[1:nv,]                       # Selektion der ersten nv Konfidenzintervallgrenzen
kfn         = kfn[1:nv]                          # Selektion der ersten nv Konfidenzintervallüberdeckungsevaluatore

# Konfidenzintervallvisualisierung
I           = rep(NaN,nv)                         # nicht überdeckende KIs für Visualisierung
I[kfn == 0] = 3.5                                # Markerpositionen
plot(1:nv, y_bar,
     type = "p",
     ylim = c(0,4),
     xlim = c(0,102),
     xlab = "Simulationen",
     ylab = "",
     pch = 19,
     cex = 0.5,
     main = "Konfidenzintervalle")
arrows(
  x0 = 1:nv,
  y0 = kappa[,1],
  x1 = 1:nv,
  y1 = kappa[,2],
  code = 3,
  angle = 90,
  length = 0.01,
  lwd = 0.7)
abline(mu, 0,
       col = "gray80",
       lty = 1)
lines(1:nv, I,
     type = "p",
     pch = 13,
     col = "darkorange")

# Testvisualisierung
plot(1:nv, phi,
     type = "p",
     ylim = c(-0.2,1.2),
     xlim = c(0,102),
     las = 1,
     xlab = "Simulationen",
     ylab = "",
     pch = 16,
     cex = 0.6,
     yaxp = c(0,1,1),
     main = "Hypothesentests")

# PDF-Speicherung
dev.copy2pdf(
  file = "../Abbildungen/grundlagen_frequentistischer_inferenz.pdf",
  width = 9,
  height = 6)
dev.off()
```

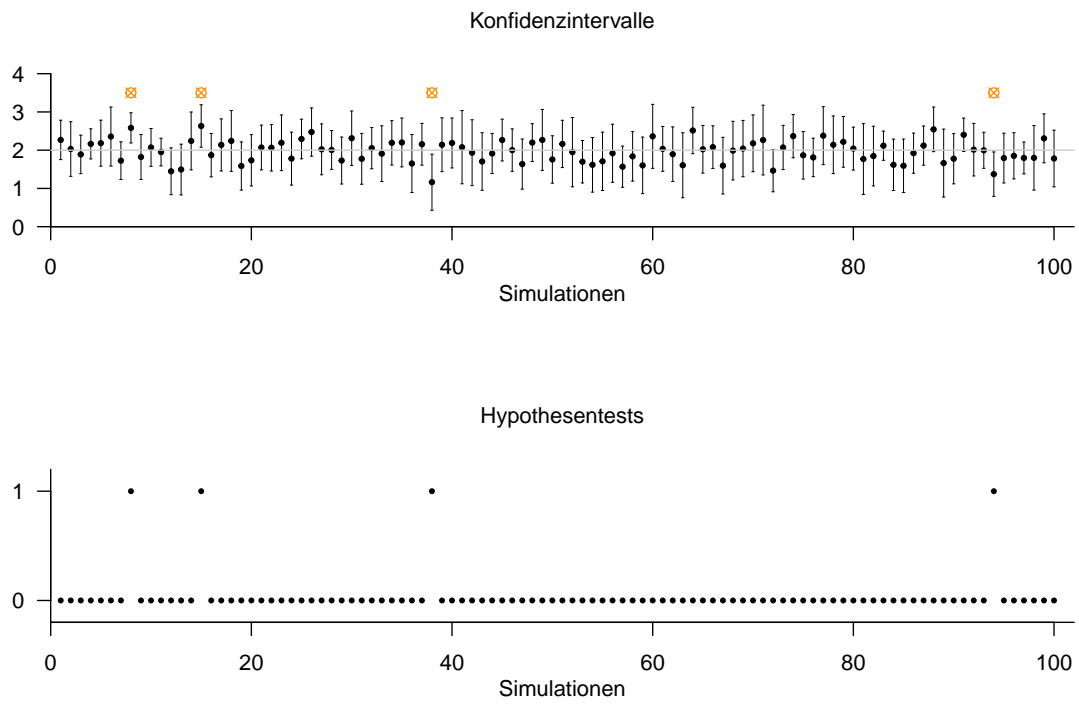


Abbildung 1. Grundlagen Frequentistischer Inferenz am Beispiel von Konfidenzintervallen und Hypothesentests im Szenario des Einstichproben-T-Tests.